£1

Coder Adventure

ISSUE 3 SEPTEMBER · 1989

SUBSCRIPTIONS

Price per single issue:

£1.00 United Kingdom
£1.75 Europe (+ S.Ireland)
£1.75 Rest of the world (surface)
£2.25 Rest of the world (air mail)

Price per 12 issues (1 year):

£12.00 United Kingdom
£21.00 Europe (+ S.Ireland)
£21.00 Rest of the world (surface)
£27.00 Rest of the world (air mail)

All payments must be made in pounds sterling. Cheques should be crossed and made payable to "C HESTER" please. (This also applies to the costs below.)

ADVERTISING RATES

£5.00 Inside full page (black-and-white A5)
£6.00 Outside back cover (on coloured paper)
£2.50 Inside half-page
£1.25 Inside quarter-page

LETTERS

Letters sent to the magazine may be chosen for publication unless marked "Not for Publication". If you require a reply other than printed in the magazine, please enclose a stamped self-addressed envelope.

COPYRIGHT

Copyright on anything which does not carry the writer's name belongs to Adventure Coder. Copyright on all other items lies with the writer as Adventure Coder does not pay for contributions. Readers whose work is printed in the magazine continue to hold copyright on all material written by them and are free to use it elsewhere.

ABOUT ADVENTURE CODER

The opinions expressed within this magazine by individual writers are not necessarily shared by the Editor. Adventure Coder does not recommend or condone any particular firm or product.

Issues of Adventure Coder come out during the middle of each month. The deadline for contributions is the 25th of the previous month.

Software and hardware is greatly appreciated for review purposes no matter what computer it is for. Adventure writing utilities are especially welcome.

Adventure Coder is produced on a Commodore 64 with a Citizen 120D printer using a variety of programs.

PUBLISHER: Mandy Rodrigues

EDITOR: Christopher Hester, 3 West Lane, Baildon, Near Shipley, West Yorkshire, BD17 5HD, England.

CONTENTS

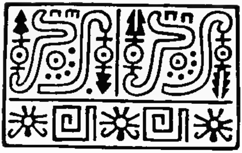
Mexican illustrations reprinted under free copyright allowance from the book "Design Motifs of Ancient Mexico" by Jorge Enciso (Dover Publications, Inc.)

Bigger and better!

Ah - the summer's ending! I can hear the sound of computers being unwrapped again, adventure games being played again, the winter months approaching once more... what better time to tackle that game you've been wanting to write all year - or maybe what better time to finish that game you've been struggling to write during the summer! Either way, the good news is this issue is the biggest yet! The increase in size is all down to you, the reader. Many, many thanks to all the people who've taken the time to write in with your articles and other contributions over the last month. Without these, "Adventure Coder" would be about five pages thin! Don't forget I'm still looking for more of your pages on whatever aspect of adventure writing interests you. Never feel your articles are too slight to be of use to anyone else - how do you know that if you keep them to yourself? The only way to get a good response from a routine you may have written is to send it in for me to print! That way, you not only achieve instant fame, but your routine will be seen by other writers, for whom it may be just what they have been trying to write themselves. You'd be surprised how one writer will have written a routine yet someone else has written the same thing in a shorter and better way, one that would never have occured to the first writer. No two minds think alike, so it's important that we all "club together" and share our knowledge of adventure writing. Otherwise the standard of home-written adventures will be lower as writers will have had to work everything out themselves, not always in the best way. So don't be shy, send me your routines! Or be forever selfish...

A handy tip has been passed on to me from George March's Mum - thank's Mum! - to do with sending in contributions. If you want to send me some A4 sheets ready for direct insertion into the magazine, then get yourself a small roll and put them inside! That way, there'll be no chance of the pages suffering from the effects of folds. An A5 envelope is otherwise okay, but any more than one fold on the sheets and the folds will show up when photocopied, maybe even chipping the letters off the page and ruining your sheets. George now sends me his PAW Prints column in a small roll, and it pastes down beautifully! Canny idea, that!

Next month sees a group of excellent PAW routines from Darren Rose, more mayhem from Matthew Conway, my own article on atmosphere, plus a host of features including... bet you can't wait... our first competition, with mega exciting prizes to be won!! Make sure your subscription spans to the next issue! And let's have plenty of articles, artwork, adverts, and anything else you wish to send in... I'm looking to expand the magazine again, let's get it to 40 pages!



Hi!

First off, well done to Chris Hester and Adventure Coder for an excellent first issue. Hopefully the adventure writers of this world will spot this great little magazine and keep it alive and kicking for a very long time. I for one will be contributing on a regular basis (time permitting!).

There is, however, one thing I would like to point out to readers. And that is the listing of Mastertronic's address in the Useful addresses section. If you, like myself, program using PAW or another adventure writing aid then don't bother sending anything to them because, for some reason they seem to think that we are "cheating" by not learning machine code!!!

Last but not least a little message to a programming mate and maybe a game for the Whatever happened to..? section. Mark - where's "Possessed" you've been promising you'll finish (or is that start it!?) for the last four or five years!

Best of luck,

Shaun Allston, 61 Goldcrest Road, Ipswich, Suffolk, IP2 0SF.

P.S. If any readers need any help with writing with PAW then please feel free to contact me at the above address enclosing a (yes you guessed it!) SAE.

I get the feeling he likes the magazine! I wonder if Mark is the same programmer I know who should have his spoof adventure "Inbanana Jones" out by now, surely? (He's also supposed to be starting to playtest a game I had published last year!!) As for the Mark that Shaun knows, he tells me he'll finish "Possessed" as soon as he can afford a Rampack for his ZX81, okay?

Dear Coder,

The select band (perhaps a little too select) of people writing what might be termed standard single player adventures are what 'Coder' is all about and for. But in view of the fact that lack of readers was one of the factors that led to the demise of 'Adventure Contact', the predecessor to 'Coder', perhaps it should expand it's ambit to also include in it's brief Multi-User Games (MUGs), Role-Playing Games (RPGs) and even Play-By-Mail (PBM) games, particularly computer and hybrid (that's computer plus human) moderated ones, as it would not be unreasonable to consider that these fall at least loosely into the fold of adventure type games. The likes of games like "Tir Na Nog", "Dunn Durrech" and "Mereport" also surely qualify to be included in this broader remit.

Might I even go as far as to suggest (probably to cries of "sacrilege") that since there is no forum for writers of other types of games, these could be given a chance to contribute to 'Adventure Coder' (perhaps then renamed 'Games Coder') as a (somewhat radical I agree) way to expand this enterprise to a viable size.

Another radical suggestion I have is that moves should be made towards the formation, under the auspices of 'Coder', of a consortium/association of independent adventure producers, if for no other reason than to turn several small voices into one larger one to promote our cause.

On the subject of MUGs, you may have seen details of the launch of "Bloodstone", a revolutionary new multiplayer telecommunicative adventure game. From what I have read of it this would seem to embody the principles for a 'participative fiction' game using fully described objects and rule based enaction of commands, as laid out in my article on 'Interactive Fiction' in issue 14 of 'Adventure Contact'. This game implements fully described objects by way of a component object hierachy, where nearly all objects are made up from several sub-objects (many themselves similarly built up from yet other components) to the extent of say a rose bush having individual petals to flowers, leaves and thorns on stems and branches. Whether it has rule based as opposed to imperative actions I do not know. I don't think it has gone online yet, but as I don't have a modem I can't play it even when it does, I would be enormously grateful if anyone who does get to play it to write in with a review at the earliest.

Further on unreleased items from Gilsoft, Tim Gilberts was I understand working on a 'PAW Advanced Users Guide' (or something of that ilk), which was to be of the form of generalised answers to the most frequently asked questions of the "how do I go about doing such-and-such" type. The role and contents of this was to have transferred to 'The Forge' (Gilsoft's own aborted attempt at filling the shoes of 'Adventure Contact', not the coincidentally named and similarly aborted effort of Shaun Allaton). Presumably the contents for this are lying in some wordprocessor file, and it is my hope that either Gilsoft can be persuaded to either publish the book(let) as originally envisaged or contribute it's contents to the pages of 'Coder'.

Gerald Kellett, 28 Queen Street, Stamford, Lincolnshire, PE9 1QS.

Since an increasing number of adventure games are now incorporating RPG elements, I welcome any material on this subject - infact Matthew Conway has already agreed to write RPG-related stuff for me. Check out his 'A-Z of RPG' in this very issue! As for PBM games, I can't see how they directly link to computer adventures in a way that is of use in this magazine. You either play a home computer game or you fill out turnsheets to play a PBM. Having said that, a lot of PBMs are much like adventures in their gameplay (fantasy lands, magic, etc) and 'Adventure Probe' once had a PBM column written by Tony Collins, which I confess I enjoyed while it lasted. I'll throw the ball out to your courts - anyone fancy writing a PBM page?

"My comments on 'Adventure Coder'... being interested in adventure writing it was just what I wanted. The articles were well written and informative (many fanzines think they can sacrifice good grammar and 'spelling' (etc)). I especially liked the GAC+ review as I had heard of it, but it seemed none of the professional magazines had!" - Mitch Pouirat, MSB Games, Cheshire.

"Thanks very much for my second issue of Adventure Coder. I am very impressed with it... I found the article about STAC especially informative." - Sue Medley, Kent.

"I am always willing to learn and although I do not use any of the commercial utilities I am sure I will find something of interest in Adventure Coder." - Tom Frost, Tartan Software, Montrose.

The A-Z Of RPG

By Matthew Conway

RPGs, or role-playing games if you didn't already know it, have their own peculiar lingo which seems to put many people off what is otherwise a most fascinating hobby. If you are one of those people, here's a glossary of the most common terms along with a layman's definition which should translate into whatever language you speak.

Adventure: a torturous experience devised by your GM designed to deliver the maximum possible ego boost when you fail to complete it and he delivers a step-by-step guide as to what you should have done.
Attributes: the building blocks of your character which show exactly how weak he is compared to everyone else's.

Campaign: an ongoing series of adventures revolving around one thing, generally the GM's twisted sense of fun.
Character: any being capable of attacking, stealing or generally making someone else's life as unenjoyable as is possible.
Character sheet: the piece of paper you always forget to bring with you to gaming sessions.
Combat: the simplest way a GM can kill your character. Trolls usually featuring quite frequently.
Critical: a generally nasty attack which the GM has decided will cut your character into two because he didn't like the way you rolled the dice.

Dexterity: an oft-encountered attribute showing just what a klutz your character is.
Die (1): the correct singular of dice but encountered about as frequently as a compassionate and humane GM.
Die (2): what your characters will tend to do quite frequently.
Dungeon: a convenient place to hold an adventure due to its being underground, dark and thus quite suitable for ambushing unwary characters.

Encumbrance: how much your character is carrying, generated by counting the number of items this amounts to and then multiplying by your shoe size.
Enemy: the other characters in your party.

Fumble: a chance for the other players to have a good laugh as your character's attack turns into a determined attempt to decapitate himself.

Games Master: the nice man who runs your adventures for you and to whom you had better be really polite if you ever want to see your character get on in the world.
GM: a common abbreviation for Games Master used by those who have no respect for such exalted personages and, for some strange reason but totally unconnected, also seem to lose more characters in a game than anybody else.

Health: an attribute showing approximately how long it is until your character dies but rarely completely accurate due to acts of God (like accidently standing on the GM's foot).
Hits: the number of times the GM has decided that he doesn't like your character.

Initiative: something your character lacks most of the time.
Intelligence: see Initiative.

Level: a numeric representation of how close you are to being able to insult your GM without him being able to kill your character in one stroke.

Melee: a term which allows your GM to kill your character and practise his French at the same time.

Non-player character: a character the GM decides is going to try and kill yours.
NPC: a type of glue produced from horse bones, but also see Non-player character.

Parry: a vain attempt to stop someone slicing your character in half by jabbing your character's puny dagger in the same direction as his enemy's two-handed battle-axe.
Party: your character plus five others all out for his blood.
Patron: a small adventure idea found in numerous Traveller supplements which saves the GM the bother of having to create an adventure for you.
PC: a member of the constabulary, but also see Player character.
Player: either a masochist or someone with a serious death wish.
Player character: a character run by a player who decides it is going to try and kill yours.
Profession: what your character does when he isn't getting himself killed on some darned fool adventure.

Reactions: see Initiative.
Role-playing game: a game in which one person tries to impose his ego upon five other people who are intent upon doing the same thing themselves.
Round: a period of time equal to how long it takes one Troll to kill one player character with one very large club.
RPG: a chemical which destroys ozone, but also see Role-playing game.

Scenario: the same as an adventure but a flashier name which gives players the impression that the GM knows what he is doing because he can speak Italian.
Session: as long as it takes for the players to get bored with the GM's underhand tactics whereupon everybody simultaneously gets up and leaves.
Skill: what you think your character is and what everybody else thinks your character isn't.
Solo-adventure: a convenient way to cheat because you're the only one playing.
Speed: the one thing you'll want to swap all those gold pieces for when being chased by a very hungry Troll.
Spell: a convenient device to allow self-confessed loonies to create rabbits out of thin air.
Statistics: the one thing you'll need to be good at if you want to be a GM, although a sadistic streak also comes in handy.
Strength: an attribute which seems to rise in inverse proportion to intelligence.
Supplement: yet another to the RPG you thought was only going to cost you £15.

So much for the terminology, what about the games? There are so many to choose from that it can be daunting trying to pick out the good ones from the turkeys. For your convenience, however, here is the definitive guide to fantasy RPGs...

Advanced Dungeons & Dragons: old and awkward, the originalest running to a meagre 15 or so hardback manuals has just been upgraded to a second edition which has been changed so much its mother probably wouldn't recognise it. Only advisable for nostalgic players or rules lawyers who think that searching through 15 manuals for a table detailing the weight of a chicken is fun.
Dungeons & Dragons: the granddaddy of them all and a much simpler version of Advanced Dungeons & Dragons (who would have guessed that?), this RPG seems to have been forgotten by TSR of late in the upgrading of its big brother (son?). A good way to start for beginners but anyone with more than two years' role-playing experience who is still playing it deserves to be called a sentimental old fool.

Middle-Earth Role Playing: known as MERP to its friends, this RPG recreates the world of J.R.R.Tolkien as only an imbecile could - boring, complex and unfriendly. The system itself, a simplified version of Rolemaster, could pass as a language in its own right. Generally avoid.

Warhammer Fantasy Roleplay: The newest of the big fantasy RPGs, this is backed by arguably the best adventures of all time in the shape of the Enemy Within campaign. Detailed for old-hands or simple for beginners, it too seems slightly neglected by its producers, Games Workshop this time, but things are promised.

If you prefer science-fiction then perhaps...

Paranoia: the one RPG geared towards giving the GM some fun, this one was greatly changed (for the worse) for the second edition but still remains a good laugh which is, after all, the only reason you should play it. Not for people who dislike losing six characters per adventure but certainly for every other sentient being in the pangalactiverse.

Star Trek: combines the background of the TV series with the complexity of an A-level chemistry textbook. Only buy it if you're a real Trekkie but be prepared to steal the important bits and fudge the rest. Experienced players/masochists only.

Traveller: one of the first and probably still the best, its latest form, MegaTraveller, retains the flavour of the original version and is fully compatible with some of the terrible-but-fun adventures that were written many years ago. A good solid choice.

For horror freaks...

Call of Cthulhu: based around the works of H.P.Lovecraft, the system is good and the background is excellent. The adventures ooze tension and the whole feel of the game is of the kind which sadly died a couple of years ago. If this background at all interests you, beg, borrow or steal a copy.

Or if you're a wargamer...

Warhammer Fantasy Battle/Warhammer 40,000: same game different settings, the first in medieval times and the second in the year 40,000 (wow!). Be prepared to buy about 3,000 lead miniatures to play them but endless enjoyment thereafter.

Others to look out for include superhero RPGs (DC Heroes, Marvel Superheroes, Golden Heroes), post-holocaust RPGs (Twilight 2,000) and the miscellaneous RPGs (GURPS among others). Some are okay, some are turkeys. None, it has to be said, set the world on fire but you could do worse.

There you go, then. The idiot's guide to RPGs. Though, to be honest, you'd have to be an idiot not to play them.

PARANOIA

Advanced Dungeons & Dragons

DUNGEONS & DRAGONS

FANTASY

The author of this article accepts no liability at all for slandering other people's favourite RPGs.

Transport in Adventures

by Shaun "Spud" Ailaton

A lot of games nowadays seem to implement at least one mode of transport, be it a Cab, Car, Bus or in a few cases, such as in "Sherlock", Trains. Over the coming months I'll be looking at each section and showing how it can be done using PAW.

We'll start though, with the simplest of the four – the car. First of all we'll need to set aside three locations. Locations one and two will be, let's say, an office building and the player's home, and location three will be inside of the car. I've shown the example without making it so the player has to start the car although this could easily be implemented by creating an object called a Car Key and checking that when a DRIVE TO command is entered that the player is carrying the key.

Flags

60 = The location where the car is

Messages

```
No. Description
 1  I was already in the car.
 2  The car isn't here.
 3  Climb into what?
 4  Climb out of what?
 5  I drove
 6   home.
 7   to the office.
 8  I was already here.
 9  Drive where?
10  A car stands before me.
```

Response table/Process table 0

```
CLIMB CAR    PREP IN
             SAME 38 60
             GOTO 3
             DESC

CLIMB CAR    PREP IN
             AT 3
             MESSAGE 1
             DONE

CLIMB CAR    PREP IN
             MESSAGE 2
             DONE

CLIMB _      PREP IN
             MESSAGE 3
             DONE

CLIMB OUT    AT 3
             COPFF 60 38
             DESC

CLIMB OUT    NOTAT 3
             MESSAGE 4
             DONE
```

```
DRIVE *     AT 3
            PROCESS 3
            DONE
```

Process table 1

```
*     *     SAME 38 60
            MESSAGE 10
```

Process table 3

```
DRIVE HOME  NOTEQ 60 1
            LET 60 1
            PAUSE 100
            MES 5
            MES 6
            DONE

DRIVE HOME  MESSAGE 8
            DONE

DRIVE OFFIC NOTEQ 60 2
            LET 60 2
            PAUSE 100
            MES 5
            MES 7
            DONE

DRIVE OFFIC MESSAGE 8
            DONE

DRIVE _     MESSAGE 9
            DONE
```

That's all folks' Quite short and also simple. Next time I'll be looking at implementing cabs which, in my view is just as simple as the above (notice I'm doing the simple ones first?!). By the way, if you can't wait until the next time or you have a few problems programming using PAW then drop me a line at 61 Goldcrest Road, Ipswich, Suffolk, IP2 0SF and please enclose a SAE.

The Dreamer

by Chris "letter of the month in TGM021" Hester

DREAMER ... DREAMER ... YOU KNOW YOU ARE A DREAMER ...
SUPER-TRAMP?
BUT CAN YOU PUT YOUR HANDS ON YOUR HEAD? OH NO...
!?!

PAW Prints

by George March

This issue, I'd like to answer one of my own frequently asked questions, which is 'how can I make my ideas look more professional'? Or to put it another way, 'how can I write routines that people like 'Level 9' have in their games'? So a while ago (about 2 years), I came up with a couple of routines that gave me just what 'Level 9's 'Gnome Ranger' has, an 'Again', an 'Oops', and to go with these two, a 'Find object' routine.. These 'Again' routines are more or less just a universal form of LET 33 ?, the first part of which goes at the very beginning of process 2, which is measured 'after' any commands are entered by the player..

```
* * 0 NOTEQ 33 A COPYFF 33 133 COPYFF 34 134 COPYFF 35 135 COPYFF 36 136
COPYFF 43 143 COPYFF 44 144 COPYFF 45 145 NOTDONE
```

I've used value A, in the NOTEQ action above, to be the verb number of the word 'Again', or 'Repeat', etc, so you'll be able to see that all it does is, if any command that's entered into the game is NOT equal to the word 'Again', then this routine makes a copy of that sentences verb, adverb, first noun, etc, into the seven 'blank' flags given (133-145), and then ends the routine with a NOTDONE, thus telling PAW that the flag numbers have had copies made of them, but that the game isn't affected any, because it thinks that nothing has been done!
Now this section (the business end!) must be placed at the very beginning of response, just in the same way that any LET 33 ? actions have to be placed 'before' the routines that they're transformed into!

```
* * 0 EQ 33 A COPYFF 133 33 COPYFF 134 34 COPYFF 135 35 COPYFF 136 36
COPYFF 143 43 COPYFF 144 44 COPYFF 145 45
```

And so what the routine does is, if the word 'Again' is typed, it copies the copies of the previous sentence's verb, nouns, adjectives, etc, back into the response table, and does what it's already done a second time round! Okay, but what if the word 'Again' is entered more than once? Very easy, as the ' * 0 action in process 2 only works if the word 'Again' is NOT typed, so if it is, the * * 0 action in response just copies and re-copies back into itself what's already been done, for as many times as the word 'Again' is entered!

Now for my 'Find object' routine, this next routine first finds the number of the object given in the sentence (ie, the number in flag 51, which is the object number measured by the WHATO action, in place of any underlines in routines, messages, etc) and if the number of the object is less than that of the number after that of the last object in a game (given as value B), and the location of the object is NOT not created, and is therefore actually in the game, isn't worn, is not carried, and isn't at the players present location, then the room number of the player is made to be the same as the room number of the object in question, ie, the player moves to the objects location, with sys' 68 telling the player what happens. You don't need a newline with an ANYKEY/DESC, and it takes up a tiny bit of useful memory anyway, so I haven't bothered!

```
FIND _ 0 WHATO LT 51 B NOTEQ 54 252 NOTEQ 54 253 NOTEQ 54 254 NOTEQ 54
255 COPYFF 54 38 SYSMESS 68 ANYKEY DESC
```

Sysmess 68 = 'You wander round, until you find the _ !' This next routine's nearly the same as that above, just that the object hasn't been found yet, and is thus not created..

```
FIND _ 1 WHATO LT 51 B EQ 54 252 MESSAGE 69 NEWTEXT DONE
```

Message 69 = 'Sorry, I can't find that, it must be hidden somewhere!' and stops any input with a newtext. This next bit, is just as above, but if the object is worn, if so, then sys' 29 = 'You're already wearing the _!' is printed, newlined and newtexted.

```
FIND _ 2 WHATO LT 51 B EQ 54 253 SYSMESS 29 NEWLINE NEWTEXT DONE
```

And again, though if the object is carried this time..

```
FIND _ 3 WHATO LT 51 B EQ 54 254 SYSMESS 25 NEWLINE NEWTEXT DONE
```

Also for this next one, but if the object is in the players location, with sys' 59 = 'But the _ is already here!'

```
FIND _ 4 WHATO LT 51 B EQ 54 255 SYSMESS 59 NEWLINE NEWTEXT DONE
```

But what if the player tries to 'find' any object which doesn't exist in the game, ie, anything which isn't recognised by these first four routines!

```
FIND _ 5 SYSMESS 8 NEWLINE NEWTEXT DONE
```

Well this should take care of that, as this routine should only be encountered if the object in question is NOT a real item, and would thus pass through the other four routines, if so, then sys' 8 is printed and newlined, etc, and to finish off these 'FIND _' routines, how about a little rearrangement of the verb, by making value C below, equal to the number of the verb 'FIND' in vocabulary, you'll also need to make 'FIND' have a number higher than that of the word 'SEARCH' in the vocab' table..

```
SEARCH _ 0 PREP FOR LET 33 C
```

And now onto a few chages to issue one's EXIT _ routines, for a better, cleaner, 128 version (if you've plenty of memory left that is!) Its 'calling' routine from response is more, or less just the same as that found in issue one..

```
EXITS _ SYSMESS 62 PROCESS 4 NEWLINE DONE
```

The first thing we'll need to do is to define four new objects as the names of the main compass directions (though there's absolutely no need for weights of objects and their words to be used!), for example, sys' 62 = 'Obvious exit(s) ' (note the space after 'exit(s)') to make it look a bit better, with object 4 = 'north', obj' 5 = 'south', obj' 6 = 'east' and obj' 7 = 'west', now if these four (or eight and ten if you include diagonal directions and/or up and down) directions are all taken as not-created (exept for those for the very first room visited, which are placed in the initially at table, as room 0, ie, the introduction screen, which the player cannot visit, and can only 'read'), then with a few calculations of the connections table for each room visitable, and a..

```
* * ZERO 31 LET 53 64
```

In process 1, just to get the look of things right, then in process 4 (called by the EXITS _ routine in response) we could have for example..

```
* _ AT 4 PLACE 4 0 PLACE 6 0 PLACE 7 0
```

One routine of this type would be needed for each room available, and then to end the whole process..

```
_ _ LISTAT 0 DESTROY 4 DESTROY 5 DESTROY 6 DESTROY 7, etc..
```

For as many objects as the writer decides to use as direction names! And so, if the EXITS _ were called whilst the player was in room 4, then the above routines in process 4 would give..

'Obvious exit(s) north, east and west.'

As the LET 53 64 action always places comas and an 'and' in any screen listings of objects if used!
And next, how about what I believe to be the ultimate 'Oops' routines going, as the 'Oops' routine given in the PAWS technical manual (and possibly the one given in 'Adventure Contact' number 13, by Mr. Bryan Kitts, though I've never actually read it!) makes the use of ramsave/ramload impossible, as the one in the manual uses an auto' ramsave in process 2, as an 'Oops', and stops the use of seperate ramsave/ramload routines, whereas the ones given here allow the use of both! So for the very beginning of process 1 (or 2, depending on the writers tastes!)

```
* * 0 ZERO 66 RAMSAVE
```

Which, just like that given in the manual, causes an automatic ramsave 'before' anything can be entered by the player, but only if flag 66 is zeroed! These next six routines are all for response.

```
RAMSA _ 0 ZERO 66 SET 66 RAMSAVE DESC
```

So if ramsave is typed, and flag 66 is nothing, ie, in 'Oops mode', the routine sets 66, ramsaves and describes the screen, but if 66 has already been set by the routine above, then it can't work a second time can it? And thus, if ramsave is typed whilst in 'ramsave mode'..

```
RAMSA _ 1 NOTZERO 66 RAMSAVE DESC
```

This first ramload routine only works if 66 is set, ie, ramsave has been used, it then ramloads the game, and clears 66, putting the game back into 'Oops mode'!

```
RAMLO _ 0 NOTZERO 66 RAMLOAD 255 CLEAR 66 DESC
```

Yes, but what if the player tries to ramload whilst in 'Oops mode'? Well this routine should then print message A = 'But you can't ramload if you haven't ramsaved a position first!' and newtexts the game!

```
RAMLO _ 1 ZERO 66 MESSAGE A NEWTEXT DONE
```

And for our Oops action whilst in 'Oops mode', which works automatically at the start of the game, and ramloads from the auto' ramsaved position in process 1 (or 2).

```
OOPS _ 0 ZERO 66 RAMLOAD 255 DESC
```

And this last one of our six response routines, takes care of the problem, if a player tries to Oops whilst in 'ramsave/ramload mode', with message B = 'You'll have to ramload your already ramsaved position first!'

```
OOPS _ 1 NOTZERO 66 MESSAGE B NEWTEXT DONE
```

Now to end with, here's a cute, little location finder for any objects that aren't in the same position as the player, so for response we could have..

* _ NOTEQ 33 A WHATO LT 51 B NOTEQ 54 252 NOTEQ 54 253 NOTEQ 254 NOTEQ 54 255 SYSMESS C PROCESS 5 NEWTEXT DONE

Wlth 'A' being the verb number of the word 'remove', eo that this routine won't interfere with the removel (or getting) of any iteme from e container, 'B' ie the number after that of the lest objecte number, end sys' C = 'The _ is etill ln the ', pleese note the epece after 'the', end process 5 is concerned wlth the printing of the neme of the objects container on ecreen, so for procese 5..

* _ EQ 54 D LET 51 G

* _ EQ 54 E LET 51 F

Wlth 'D' being the locetion number where the object is loceted, ie, s contelner-room number, end 'G' being the object number of the container, but if you 128'ers wlth sll thet memory, wish to make your gemes e blt more frlendly, how about, as in my second exemple, if we use 'E' ss s normel (not contelner) room number, end 'F' es the neme of the room, you would then have to uee eome objecte es the namee of theee rooms however, just ae the EXITS _ examples ebove (with no need for weights, or words of objects to be used!), euch es 'D' = room 5, 'G' = object number 5 'A cerdbosrd box', 'E' = room 6 'the kltchen', though you could elso do it like thie..

* _ EQ 54 E MESSAGE F

So, egein wlth 'E' being e non-contsiner, normel room number, end messsge 'F' belng the neme of the room prlnted, insteed of en object ueed, end to end thle proceee, we'll need e little routine like the one below, wlth 'H' = '_(', for the twln * _ EQ 54 D/E routines elready glven, which prlnts the neme of the object-conteiner/ locstlon where the object in queetion le contained!

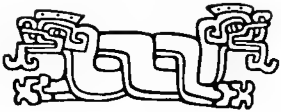
_ _ MESSAGE H DONE

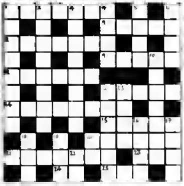
And ee elweye, you cen elther contect me here vie the 'Coder', or et..

93 ROBERTS STREET, NEWCASTLE UPON TYNE, NE15 6BE

For eny comments, complelnte, help thet you need, or hopefully whet we're looklng for ie help you can glve to your fellow wrlters! I mesn holdlng onto your own routines mey help you mske your own gemes look good, but helping othere with eome idees you've created, not only helpe them get their gemes looking greet, but should meke you feel very proud ln whet you've done to help, end lt sleo gete your neme end/or if you ectuelly publish your gemee under your own lebel, your compeny's neme in prlnt, with full credit! So go on, give e little!

So good bye 'till next month!





Crossword

by C H

ACROSS

1) Observed, an uncoated cake? (7)
7) Wet software house (5)
8) Criminal (7)
9) The key to dreaming (5)
11) Heavenly (6)
12) Prevent (5)
14) Take over a new country (8)
15) Where all bands play on (5)
20) Placed against - because tired? (6)
21) Splits up (7)
23) Poem (3)
24) Negative (2)
25) Total (3)

DOWN

1) Highly unlikely, mate! (2,6)
2) Someone you can't trust with secrets (4,4)
3) Window dressers, pull yourselves together! (6)
4) Tiny specks (4)
5) Not there (4)
6) The same card! (4)
10) Angered (7)
12) Value something (6)
13) Where you take a sick pet (4)
16) Smallest particle (4)
17) Where Adam met Eve (4)
18) False hair (3)
19) Immorality (3)
20) Initials for holy study (1,1)
22) Get up and -- something! (2)

Solutions to last month's crosswords

1

```
P R O B E S   G A C K E D
L     E   E   O   R     R
A     A   A D D   I     A
N O O N E   I   B E I N G
E     O N E S T A R     O
T I E   Q   T   R   R U N
    M O U S E T R A P
S O U   I   M   I   G A S
Y     T R A P P E D     Y
S C O R E   E   R E A L M
T     A   U R N   N     B
E     I   S   E   S     O
M E A N I E   O L E V E L
```

2

```
T R A N S M U T A T I O N
U   V       R       R   E
R A I N   A N D   F I N E
N   D E A R   O P U S   L
A     A   I O N   R     L
B   B R E D   E V E R   E
L O B   S       I   A S S
I   C O P Y   S E C T   S
N     V   E R A   M     T
D   S E A T   M E E T   U
E V E N   I C E   F U R S
I   N       A       N   A
E X T R A O R D I N A R Y
```

Doors in GAC adventures

by Matthew Conway

A door obviously serves a major purpose in any adventure, be it computer adventure or role-playing game, that is to bar progress until some specific action is taken to bypass it. Usually this entails the location of the correct key to unlock it, thus sending the player off on a secondary quest which can involve all manner of problems and puzzles, and this is the example which I will use. However, they have other uses not covered here: a door could be impassable but, as long as the player does not know this, still seem like a problem to be solved, thus sending him/her off on a wild goose chase for a non-existent key whilst the real path forward lies somewhere else entirely; it could be a clue in it's own right, giving information if the correct steps are taken; it could be a dangerous monster in disguise, ready to entrap the unwary adventurer which it has just fooled; and so on.

However, before using doors in your adventures, think carefully: are they important enough to make the memory expenditure worthwhile? Using the ST version of GAC where nearly 300K is available for use as well as the ability to access discs is one thing, but the humble Speccy only leaves 23K to work with, and with even the simplest of doors using about 300 bytes, I for one do not think that they are economical enough to deserve use. *(Better quit the article now then, Matthew! - Ed)* This is purely a personal opinion though, and the rest of this article is designed to help you make up your mind and, if you decide to use them, to show you how.

There are two ways of using doors in GAC of which only one is presented here because it is easier to code. The first is to use a marker to indicate whether a door is either open or closed and/or a second one if the door can also be locked and unlocked and then display the information from High Priority. This is very messy, however, and things can get very complicated indeed with the juggling with statements which can occur to get exactly the right effect - for this reason, those of you who wish to use markers are on your own because I will not be covering it here. I will, however, be covering the second method, namely the use of objects to represent which of the states of the door is currently present and swapping between these as thing change. This has a far more logical approach and is much easier to debug when problems arise.

The simplest set-up where doors are concerned is that where an unlockable door (ie: open or closed only) controls access between two rooms. Here we need to use four objects to represent that door: one in each of the rooms for the door in both it's states. This results in the following coding assuming that: the player starts in room one; the door connects rooms one and two; room two is to the north of room one; and that the player can only move between the two when the door is open.

Objects

No.	Description	Weight	Starts at
1	a closed door in the north wall	255	1
2	an open door in the north wall	255	0
3	a closed door in the south wall	255	2
4	an open door in the south wall	255	0

Local Conditions

```
Room Line Statement
 1    1   IF ( VERB [ OPEN ] AND NOUN [ DOOR ] AND HERE 1 ) 1 SWAP 2
          3 SWAP 4 MESS [ You open the door ] WAIT END

      2   IF ( VERB [ OPEN ] AND NOUN [ DOOR ] ) MESS [ It's already
          open ] WAIT END

      3   IF ( VERB [ CLOSE ] AND NOUN [ DOOR ] AND HERE 2 ) 1 SWAP 2
          3 SWAP 4 MESS [ You close the door ] WAIT END

      4   IF ( VERB [ CLOSE ] AND NOUN [ DOOR ] ) MESS [ It's already
          closed ] WAIT END

      5   IF ( VERB [ NORTH ] AND HERE 2 ) GOTO 2 WAIT END

      6   IF ( VERB [ NORTH ] ) MESS [ The door is closed! ] WAIT END

 2    1   IF ( VERB [ OPEN ] AND NOUN [ DOOR ] AND HERE 3 ) 1 SWAP 2
          3 SWAP 4 MESS [ You open the door ] WAIT END

      2   IF ( VERB [ OPEN ] AND NOUN [ DOOR ] ) MESS [ It's already
          open ] WAIT END

      3   IF ( VERB [ CLOSE ] AND NOUN [ DOOR ] AND HERE 4 ) 1 SWAP 2
          3 SWAP 4 MESS [ You close the door ] WAIT END

      4   IF ( VERB [ CLOSE ] AND NOUN [ DOOR ] ) MESS [ It's already
          closed ] WAIT END

      5   IF ( VERB [ SOUTH ] AND HERE 4 ) GOTO 1 WAIT END

      6   IF ( VERB [ SOUTH ] ) MESS [ The door is closed! ] WAIT EN
```

Now all you have to do is insert the conditions for locking and unlocking, which now shouldn't be so difficult should it

Sentinel Chart

by Chris H.

1) "Meenie Meenie" - Billy Idol
2) "One More Tree" - Phil Collins
3) "Somebody's Watching Me" - Rockwell
4) "Fun In Hyperspace" - Roger Taylor
5) "The Only Way Is Up" - Yazz
6) "Don't Give Up" - Peter Gabriel
7) "Sentry Bloody Sentry" - U2
8) "I, Robot" - Alan Parsons Project
9) "Landscape Of Confusion" - Genesis
10) "Plinth Plinth Meanie" - The Beatles

DIGITAL DYNAMITE

54 Watermill Road, Fraserburgh, Grampian, Scotland, AB4 5RJ

**A Great Opportunity To Make Money
From Your Commodore 64 Programs And Music**

Have you written a program which you feel is good enough to be making money for you? If you have then DIGITAL DYNAMITE can help you.

An Exciting Opportunity

We are looking for ANYTHING at all, be it Games; Utilities; Demos or even Music. If you've programmed it then we want to see it. If we accept it, *and we won't turn down anything which is of a reasonably high standard*, then you are on your way to MAKING MONEY.

Why Digital Dynamite?

We know how difficult it is trying to have your program commercially released. Very few companies accept utilities. Virtually none accept demos. Games are expected to be of an unreasonably high quality. GREAT GAMES are turned down, often in favour of those with poor playability but attractive graphics.

We, like you, are part of the great games buying public. We KNOW when a game is fun to play and we *will NOT be swayed by fancy graphics*.

We want your programs as part of a collection which we will sell as a package on one disk or tape. *The enjoyment gained from having many good programs in one package is greater than that gained if all these were sold as separate packages*

Why Will It Sell?

Here's why:

. Most games are advertised for a month and are in the shops for about two months, we will sell by mail order, ensuring a much longer advertising and selling period:

. There will be approximately sixteen programs on one disk or tape, roughly ELEVEN will be high quality games while the others will be utilities and demos;

. The package will sell for the same price as most full price tape based games and for LESS than most full price disk based games:

. Advertising in, for example, Zzap! magazine reaches approximately 150,000 game enthusiasts every month:

. There will be something in the package that nearly EVERYONE can enjoy or make use of.

What Programs Are We Looking For?

Basically, anything:

. Graphics Adventure Created adventures;
. QUILL'd adventures;
. Adventures containing all your own programming, even in BASIC;
. Role Playing games;
. Strategy War games;
. Football Manager type games;
. Arcade Action games;
. Arcade Adventure games;
. Platform games;
. Shoot-'em-ups created using the Shoot 'Em Up Construction Kit;
. Shoot-'em-ups which are all your own work;
. Pinball tables;
. ANY other type of game;
. ANY utility program (e.g. Sprite Designers, Music Creators, etc.);
. DEMOS which are highly original;
. MUSIC.

We only ask that the program MUST be ALL your own work.

What About Music?

Don't worry about music for your games or demos. In most cases we can arrange to have great music incorporated into your programs for you, giving that final professional touch.

What's In It For You?

You will feel *successful* knowing that a program which *you wrote* is a major reason why people are buying a software package. You will gain *satisfaction* from the knowledge that people are using and enjoying a program which you have created. Also, you will receive *payment* for a program which, possibly, hasn't seen the light of day for quite a while.

All money will be paid in royalties. This means that, for every disk or tape which we sell containing your program, you will be paid a proportion of the profits. Although the total amount of royalties paid for each tape or disk sold will be quite a large percentage, this will be divided proportionally between all of the contributors.

Using a medium sized good quality program as an example, a contributor can expect to receive approximately £20-00 for EVERY 100 disks or tapes sold.

Should we achieve as few as 1000 sales then this contributor would earn themselves £200-00, which is great for a program which would otherwise be *gathering dust amongst their tape or disk collection.*

The best selling titles can achieve well over 20,000 sales. At this level of sales, the above contributor would have earned FOUR THOUSAND POUNDS. This contributor could be YOU.

Our Guarantee To You

We GUARANTEE that you will be paid promptly EVERY MONTH while the package containing your program sells. We are not like some other softwara companies, we believe that our contributors are the MOST important peopla and regular payment IN FULL ensures that they will continue to send their programs to us in the future.

Go For It

Sand us your programs today. WHAT DO YOU HAVE TO LOSE?

Our first package will be ready very soon, so DON'T DELAY, POST TODAY and your program could be on it.

You don't even naed to spend ages writing the instructions. All wa require is brief, clear information so we can assess the program fully.

Send your programs to: Keith M^{c}Leman
Digital Dynamite submissions
54 Watermill Road
FRASERBURGH
Grampien
Scotland
AB4 5RJ

Remember, your talent and your hobby could be making you money and all for the price of a postage stamp. POST YOUR PROGRAMS NOW

[illegible]

Keith M^{c}Leman

Partners in Digital Dynamite are: Russell Barbour.
Keith M^{c}Leman

A Whole STAC Of Problems

By Matthew Conway

Part The Second: What You See Is What You Get

Right then, you've read my article on getting the most from Special Condition 17 and so now your adventure opens with a beautifully-designed title page which draws the player into the game immediately. He marvels at its aesthetic value, reads the on-screen messages, presses a key to access the adventure proper and...

Good god! What's that? The artistic title page vanishes to reveal a poorly-presented, semi-illegible location description which immediately destroys the captivating atmosphere you strived to attain. The player's estimation of your adventure suddenly drops by 1000% and he wonders if he hasn't bought a turkey after all.

Well, okay, maybe a slight exaggeration there, but the point remains valid. Having slaved away to produce a perfect introduction, there's nothing worse than showing your slackness by leaving the player with a first location description which leaves a lot to be desired. The point of this article is to try to help you avoid this easily-found pitfall and instead give your adventure a much better presentational look.

In fact, this can be achieved very easily with what are, generally, only small changes. There is one large routine, which I'll show you in a moment, but even this is relatively straightforward and easy to implement.

The shorts first, though:

1) Always make sure that you place a <u>clear-screen</u> character at the start of room descriptions, both long and short. This is achieved by pressing <u>Shift-Help</u> followed by <u>Shift-Home</u> and results in the locations being printed at the top of a blank screen - much neater than having to search through piles of text just to find where you are.

2) Similarly, always add a <u>Control-J</u> followed by <u>Shift-Help</u> and <u>Return</u> at the start of the <u>You can also see</u> message because this makes the list of available objects more conspicuous and saves the player having to search through the end of the location description for it.

3) If at all possible, use 80-column text because this halves the depth of any text printed on the screen, consequently making it much neater. It also cuts down on the amount of untidy scrolling which takes place. However, it's better to use 40-column text with a legible font than 80-column text with an illegible font so be careful.

4) Always use a cursor of size 7, ie: don't tamper with the <u>cursor</u> command: a solid square is easier to follow than a thin line.

5) Finally, always check the message or location text you have entered by seeing what it will look like in the actual adventure. STAC can and will print brackets and quotes at the end of one line and the text they enclose at the start of the next and nothing can be worse than seeing this type of error. If necessary, juggle with the text so that everything appears on the same line.

Right, now that that's over, I'll finally reveal the routine which I promised you earlier in this article and also in the last issue of Adventure Coder. It is quite long-winded for what it does, but it's simple enough when you think about it and it does the job well so I'm sticking with it! Anway, what it does is to smarten up the results of the <u>list</u> command. Normally, this just prints all the objects in a room with commas between each pair and leaves you to tack a full-stop on at the end. Unfortunately, things like You can also see a sword, a shield, a lamp. don't smack of particularly good English. Wouldn't it be nicer if, instead, the message on the screen was You can also see a sword, a shield and a lamp. and the full-stop was automatically added? Well, here's the routine that does just that.

```
Messages
1     ,
2     and
3     .
9913 {DOWN}{ CR }You can also see
9916 You are carrying
9917 nothing.
9927 It's pitch black. You can't see a thing.

Low Priority Conditions
      if verb "i" then special 19 wait

Special Conditions
17    1 mess$ 1 2 mess$ 2 3 mess$ 3 added to the start
14    if set? 1 and reset? 2 then message 9927 return
      descing room
      if zero? firstob room then return
      repeat
      firstob room to 9999
      until zero? firstob room
      message 9913
      repeat
      ( ( firstob 9999 ) + 9000 ) mess$ 0
      if cntobj 9999 = 1 then 1 add$ 0
      if cntobj 9999 = 2 then 2 add$ 0
      if cntobj 9999 > 2 then 3 add$ 0
      print$ 0
      firstob 9999 to room
      until zero? firstob room
15    if set? 1 and reset? 2 then message 9927 return
      set 0
      if visit? then descsht room draw pictof room else descing room draw
            pictof room
      visit
      as for Special Condition 14 from the first 'repeat' to the end
19    if zero? firstob with then message 9916 message 9917 wait
      as for Special Condition 14 from the first 'repeat' to the end except
            that all references to 'room' should be changed to 'with', the
            reference to 'message 9913' should be changed to 'message 9916' and
            a 'wait' should be added at the end
```

Phew! That isn't quite it yet, though. In addition to the above, you'll also have to enter the short object descriptions into the message table as well as into the object table: these should occupy positions 9000 greater than the object number, so if object 1's short description is a sword then message 9001 would also be a sword and so on. Make sure that these descriptions are 30 characters long at the most, though, otherwise you'll start getting characters cut off the end!

I suppose you want to know how all this works? Oh, very well then. The addition to Special Condition 17 sets up strings 1, 2 and 3 to contain the same text as messages 1, 2 and 3 respectively, ready for when a room description, look command or inventory command is entered. Note that the line in Low Priority Conditions for dealing with inventory commands replaces the two given in the STAC manual and directs the whole routine to a new Special Condition, number 19, so that loops can be used.

For an example, we'll look at what happens if the player types look. A Low Priority Condition passes control to Special Condition 14. If the room is dark and the player has no source of light then the message It's pitch black. You can't see a thing. is printed and that's that. If the room is lit in some way, however, the long room description is printed and the routine really begins to do its job. If there are no objects in the room then, again, that's that and the routine is exited. If there are some, however, they are all moved to room 9999 which must not be used for any other purpose whatsoever unless you like your adventures to work illogically. Then, the message You can also see is printed on a new line. What is printed on the screen next depends on how many objects are in room 9999. If there is only one then its short description is printed followed by a full-stop; if two then the short description is printed followed by and because the next object must be the last; if three or more then a comma is added. Finally, objects are moved back into the player's location one at a time until none are left and the routine ends. The same general thing happens when a room is described upon entry or an inventory is acted upon.

Looking at the listing, you may wonder why it is necessary to bother with strings at all and not simply tack messages onto the end of object descriptions. This is because STAC's willingness to put commas at the start of lines rears its ugly head and the only way to combat this is by placing the object description and the punctuation in the same message, hence the add$ which does just this.

By the time you've typed all that in, you may think that the result is pretty minimal and, ordinarily, I would agree with you. However, when you consider what you can do with STAC and the unlimited memory you can make use of due to being able to link files, it should be obvious that it is the little things which add together to make a big difference. There's no reason why you shouldn't try to tidy up all of STAC's inadequacies and, in doing so, you may stumble across a piece of code which proves to be invaluable. I have on many occasions and I hope to able to share a few more of them with you in the future.

Anyway, that's it for this article. If any of you out there have found a quicker way of doing what this article does, send it to me immediately! It could save a lot of people a lot of typing and sanity... Of course, if anybody wants to get in contact with me because they have a useful routine, a question or merely a comment to pass on, please do. Just remember that an SSAE is necessary if you want a reply outside of these pages.

Matthew Conway, 1 St George's Terrace, Station Road, Lambourn, Berks RG16 7PW

Back Issues

These are available at the same price as a normal issue (see back page for full details about prices).

ISSUE 1 July 1989

GAC+ review! PAW Prints! Machine Coding your adventures Part 1! Whatever happened to... "Valley Of The Source"! GAC graphics article on Colour, Perspective, Ellipses and Rectangles etc! Fiction - "The Burning Man"!

ISSUE 2 August 1989

GAC pokes! GAC+ pokes! Two crosswords! PAW Prints! The Ultimate Guide To Gacing! Machine Coding your adventures Part 2! Whatever happened to... sound-only games! STAC - Special Condition 17 etc! The Adventure - how to write one! Updated utilities list! Updated useful addresses!

Trans' 'Time-Tunnel' holidaya praaent..
(courteay of Geordie)
The top-13 'Have-it-away-day' weekend breaka for 3
Male/ Female/ Other (delete where applicable)

1) FRANCE: Thrill upon thrill to the glory of Madame Guillotine hacking her way through the decadent wimpa of frog nobilty!

2) RUSSIA: Unmaak your own diaaidenaki, aa you playaki a K.G.B versionaki of 'Jamaski Bondaki' for tha weekendaki!

3) SOUTH AMERICA: Wince with curioaity aa the aged, balding conquiatadorea pillage and aacrifice the friendly nativea!

4) CHINA: Join chairman Mao himaalf, at the head of the glorioua 'long march' tour of the graat-wall (graffitiata pleaaa try the 'New-York aubway' tour inataad!)

5) AUSTRALIA: Aid the criminal aettlera in decimating the indigenoua 'Abo population'!

6) ENGLAND: Cringe in tarror of tha poll-tax under the governmenta Thatcherite jackboot during the lata 1980'a. Join a wonderoua E.T acama to get the atill unemployed off the dole figuraa!

7) SICILY: Play tha god-father, waging your own peraonnal vendetta againat thoaa noaey, wop neighbora, the Capone family!

8) GERMANY: 'Saig Heil' and do your impreaaiona of the 'funny walka' with the reat of tham, aa you attand a genuine mid' 40'a Nazi-rally, and then onto a tour of the gaa-chaabera (1-day only, autograph huntara welcome!)

9) SPAIN: Swoon in glorioua horror, aa you aweat it out on a mid' twentieth century package tour of the 'Coata del rippoff'!

10) AMERICA: Gaap as north and aouth beat aach other to a pulp in thia very uncivil war, before your very eyea!

11) SWEDEN: 'Hurdy Gurdy' it with the reat of the inarticulate, incoherent, inimical and other words beginning with 'in', foreigners, as you lace into the oppoaition in this lavish Norse 'V' Viking apectacular!

12) JAPAN: get your own back on the 'loadsamoney' Samurai butchers, as a Ninja, freeing the lowly Chinese farmers and fishermen from oppression!

13) SOUTH AFRICA: Lend a hand to the whities in enslaving the peaceful African-people in their own country, or join the 3-mile queues for second-hand, infected groceriea, as a so-called revolutionary African under the 'fascist' oppresion of P. W. Botha!

Machine Coding your adventures Part 3

by Paul Brunyee

Welcome to a further article focusing on an alternative form of adventure creation. By 'alternative' I wish to describe an alternative to the current crop of adventure creating utilities such as the Quill, GAC and PAW, to name but three. Certain utilities have, by and large, acquired bad reputations from jaded reviewers due to a huge influx of similarly conceived adventures limited in varying degrees in their design. These design shortfalls can be divided into the two addressable areas of the atmosphere projected from the text, and possibly graphics, and the adventurers interface for communication - the input and parsing routines. There have been many adventures criticised in the past for having inflexible parsers and limited vocabularies but little seems to have been done to remedy this.

Certainly the likes of Level 9 and Magnetic Scrolls have developed very sophisticated parsing routines, and more recently adventure creating utilities now provide quite extensive facilities. As stated, however, I am offering an alternative to these utilities in the form of assembly languages which, to my mind, allow the greatest scope for flexibility and originality.

I would urge people not to discount assembly languages for their so called 'complexities'. As with any programming language, it can be easily grasped given time and patience. Even with the adventure creating utilities, a programming language of sorts still must be learnt, albeit at a higher level than with assembly languages.

The coding I will present concerns the topic of command parsing, or that part of an adventure which attempts to 'understand' your typed instructions. Parsing covers the analysis of sentence structures and may not only concern 'keyword recognition' but also the grammar and semantics of the sentence. Grammatical parsing is concerned with the correctness of the sentence with regards to the laws of grammar. For example, the sentence "I'm going a walk" is more accurately represented as "I'm going for a walk". Semantics cover the actual meaning of sentence content to indicate whether or not they actually make sense according to rules already defined. For example, you could try to "EAT A SANDWICH" but not "EAT A HOUSE".

Parsing has been developed in certain programs to very fine levels with the main objective being able to fully 'understand' human speech, thus allowing a most natural human interface to a computer program, which in our case is the advanture.

The parsing I will deal with concerns itself mainly with 'keyword recognition' and attempts to build a picture of what the adventurer is trying to do. Before starting coding, it is important to appreciate sentence structure and how sentences are composed. For our purposes, sentences will comprise of directives, or actions, which may or may not operate with the immediate environment. Thus at a most basic level we have the instantly recognisable VERB and VERB NOUN formats. As the adventurer wants to describe his or her actions more explicitly, we must introduce scope for further word types. For example, how would an adventurer differentiate between two keys if one was brass and the other rusty? Singularly, either key may be referred to simply as "key", but when either key is in close proximity to the other, the adventurer would have to qualify his or her actions with an adjective of either "brass" or "rusty". In a similar manner to how nouns may be qualified, verbs may also be qualified with the use of adverbs.

Additionally, prepositions are used to further clarify actions. For example, in order to give a scroll to an innkeeper in a room full of people, you could specify "GIVE SCROLL", but a more accurate statement would be "GIVE SCROLL TO INNKEEPER". However, a point to note would be the situation where you meet the innkeeper on a deserted lane. In this instance "GIVE SCROLL" should suffice for the exchange to take place. This is where a certain 'intelligence' can be introduced into the adventure, but I digress slightly - back to the coding!

To start with, I will consider a simple VERB NOUN format but in doing so will show how to cater for further word types by introducing modularity.

I will represent all word lists in lowercase ASCII fashion with each word separated by an asterisk, and each list terminated by a value 255. Using the asterisk as a word separator is not a very efficient method but is adequate for demonstration purposes. An alternative method is to implement a bit 7 overpunch (a *what!?* - *Ed*) for the last character in each word. Because each ASCII character takes a value less than 128, the high order bit in each associated byte is zero. The bit 7 overpunch involves setting this high bit to one, thereby having the character and end-of-word marker stored together in one byte. Note also that although the code will operate on the first four characters of each word, I have included the whole word in the lists so they may be referenced and used to construct 'intelligent' responses. This reduces the incident or duplicated data where you may have pairs of lists such as;

"EXAMLOOKTAKEINVEUNLO..." together with,

"examine*look*take*inventory*unlock..."

The command to be parsed is held in continuous memory locations as uppercase ASCII values, and is terminated with the value 13. This area to be parsed may be created by the command input routine detailed in Part 1 of these articles, or even by a simple BASIC program poking the characters forming the command directly into memory.

To briefly describe the operation of the following code; The body of the routine is labelled PARSE and for each word type issues calls to the label COMPAR in which all of the word comparisons are made. Upon entry to the COMPAR routine, register pair DE points at the start of the word list being examined and the two byte area POINTA holds an address of where to place the word number, if one is found. For example, if "take" is third in the list and the word "take" is present in the command, the value three will be placed in the address contained within POINTA. Throughout the code, register pair HL is used as a pointer to the command area, and more specifically, to the actual word being examined. The command area is scanned for the first word and the compare routine is called for each word type implemented. The register pair HL is then bumped along to the next word in the command area (if one exists) and the process is repeated. Comments included alongside the code should complete the description of exactly what the code does.

```
;Command Parsing Routine
;

        ORG  60000          assemble machine code at 60000 onwards
        ENT  60000          entry point for Assembler execution
;
PARSE   XOR  A              zeroise reg. A
        LD   (VERB),A       zero verb flag
        LD   (NOUN),A       zero noun flag
```

```
;       etc...
        LD    HL,COMMND     pointer to command area
CKVERB  LD    DE,VERB       address of verb number
        LD    A,(DE)        retrieve value if found yet
        CP    0             has a verb been found yet?
        JR    NZ,CKNOUN     yes, ignore verb check and branch on
        LD    (POINTA),DE   save address of flag for this word type
        LD    DE,VBLIST     point DE at the verb list
        CALL  COMPAR        call compare routine
        CP    255           have I reached the end of the command?
        RET   Z             yes, return to main code (or BASIC, etc.)
CKNOUN  LD    DE,NOUN       address of noun number
        LD    A,(DE)        retrieve value if found yet
        CP    0             has a noun been found yet?
        JR    NZ,CKEND      yes, ignore noun check and branch on
        LD    (POINTA),DE   save address of flag for this word type
        LD    DE,NNLIST     point DE at the noun list
        CALL  COMPAR        call compare routine
        CP    255           have I reached the end of the command?
        RET   Z             yes, return to main code (or BASIC, etc.)
        CP    0             did I find a noun (any)?
        JR    NZ,CKVERB     yes, loop back because HL already bumped along
;
CKEND   LD    A,(HL)        find current character in command area
        INC   HL            increment HL anyway
        CP    13            have I reached end of command yet?
        RET   Z             yes, return to main code (or BASIC, etc.)
        CP    32            have I found a space?
        JR    Z,CKVERB      yes, so loop back and check this next word
        JR    CKEND         keep bumping HL along to next space
                            or end-of-input
;
;Compare code...
;
COMPAR  LD    B,4           prepare for upto 4 characters
        LD    C,1           use register C as word count
CKSPAC  LD    A,(HL)        obtain current character
        CP    32            is it a space?
        JR    NZ,COMLP2     no, must be at start of word so branch on
        INC   HL            bump along pointer
        JR    CKSPAC        loop back
COMLP2  PUSH  HL            save start position of current word on stack
COMLP1  LD    A,(DE)        fetch current character from word list
        CP    42            is it an asterisk?
        JR    Z,EOWORD      yes, go find another word
        SUB   32            change value to represent uppercase chars
        CP    (HL)          compare accumulator with character pointed
                            to by HL
        JR    NZ,UNEQAL     if unequal, take the jump
        INC   DE            increment pointer to word list
        INC   HL            increment pointer to command input
        DJNZ  COMLP1        repeat above process for upto 4 times
;
;Found match, point HL at next word...
;
HLTIDY  POP   DE            retrieve HL from stack but want to keep
                            current HL
HLLOOP  LD    A,(HL)        what is HL pointing to now...
        INC   HL            increment HL anyway
        CP    32            is it a space
        JR    Z,EQUAL       yes, branch to label EQUAL
        CP    13            how about the end of input marker
        JR    Z,EOCMD       yes, branch to label EOCMD
        JR   HLLOOP       loop back (until HL repositioned)
;
EOWORD  LD   A,(HL)       what is HL pointing at now...
        CP   32           is it a space?
        JR   Z,HLTIDY     yes, need to update HL so jump to HLTIDY
        CP   13           how about the end of input marker?
        JR   NZ,UNEQAL    no, jump to UNEQAL to find another word
        POP  DE           remove HL from stack, but don't corrupt
                          current HL
EOCMD   LD   DE,(POINTA)  obtain address of where to put result
        LD   A,C          reg. C is the word count
        LD   (DE),A       save value of C in address pointed to by DE
        LD   A,255        set end-of-command flag
        RET               return to caller
;
EQUAL   LD   A,C          reg.C is the word count
        LD   DE,(POINTA)  obtain address of where to put result
        LD   (DE),A       save value of C in address pointed to by DE
        RET
;
UNEQAL  LD   A,(DE)       retrieve current char. from word list
        INC  DE           increment pointer anyway
        CP   42           was DE pointing at an asterisk?
        JR   NZ,UNEQAL    no, loop back until find one
        POP  HL           retrieve pointer to start of current word
                          of command
        INC  C            increment word count
        LD   A,(DE)       lest check for end-of-word-list condition
        CP   255          found end of list marker?
        JR   NZ,COMLP2    no, rejoin compare loop
        XOR  A            set reg. A to zero
        RET
;
POINTA  DEFW 0            define a 2 byte area (word) set to 0
;
VERB    DEFB 0            verb flag
NOUN    DEFB 0            noun flag
;etc...
VBLIST  DEFM "take*drop*examine*open*climb*read*go*give*"
        DEFB 255          end of list marker
;
NNLIST  DEFM "painting*chest*bottle*door*mirror*vial*pearl*axe*"
        DEFB 255          end of list marker
;
COMMND  EQU  <whatever>  address of command input area
```

As the code stands, the word lists above must contain an asterisk as the last character of the string, followed by a value 255. The command input area must also be terminated by a value 13. This was set in the sample input routine detailed in Part 2 of these articles.

Although this code only caters for verbs and nouns, the introduction of modularity can easily expand it's abilities. The code between the two points marked "A" and "B" can be duplicated for as many word types as you care to use. The code simply needs to be repeated, and the appropriate pointers from the surrounding code need to change to cause flow through this code, and then with a couple of new fields towards the end of the code you will have introduced scope for adverbs, or adjectives and the like. The code as it stands contains no references to movement commands, such as "north", "south" and so on. These may be included in the verb lists, but also as a new word type as just described. You duplicate the code from "A" to "B" and add a new field such as: MVLIST DEFM "north*south*east*west*" and so on. What could be simpler?

To demonstrate this modularity, and also introduce the feature of conjunctions, I have prepared the following code which should be inserted after point "B". This code makes use of a further word list which contains the conjunctions "and" and "then".

```
CKCONJ LD   DE,CDNJ      address of conjunction number
       LD   (POINTA),DE  save address of flag for this word type
       LD   DE,CJLIST    point DE at the conjunction list
       CALL COMPAR       call compare routine
       CP   255          reached end of command?
       RET  Z            yes, return to main code (or BASIC, etc.)
       CP   0            found any conjunction? (if so, reg. A > 0)
       JR   Z,CKEND      no, branch on
;Transfer commands...
       LD   DE,CDMMND    point at command input area
ANDLP1 LD   A,(HL)       find character being pointed to by HL
       LD   (DE),A       save it at address held in DE
       CP   13           was it the end-of-input marker?
       RET  Z            yes, return to main code (or BASIC, etc.)
       INC  DE           increment pointer to command area
       INC  HL           increment pointer to words past conjunction
       JR   ANDLP1
;
CDNJ   DEFB 0            conjunction flag
;
CJLIST DEFM "and*then*"
       DEFB 255
```

A further necessary code change would be to the JR NZ,CKEND within the CKNOUN part. This would have to be changed to JR NZ,CKCONJ. The CONJ flag would also have to be zeroised upon entry to the routine.

If either of the words "and" and "then" are found in a command, parsing is halted and all of the characters in the command to the right of the conjunction are shifted left to occupy the space of the words just parsed.

This facility can easily be accommodated in the main code by, for example:

```
        LD   A,(CONJ)     conjunction flag
        CP   0            was a conjunction found at last parse?
        CALL Z,INPUT      no, ask user for a new input
        CALL PARSE        call parsing routine
        etc...
```

This draws to a close my writings on parsing routines. I hope this code has demonstrated several features of assembly languages which you find useful and I also hope to have shown the need (that I see) for more "user friendly" systems. At first glance, an assembly language program can look horrendous, but with time you soon begin to appreciate their usefulness and how you can have complete control over the coding of your ideas.

Investigating PAW

by Gerald T Kellett B.sc

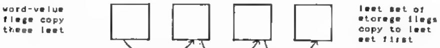
The significant changes to PAW have been the additions of; the user overlays, the multiple PARSE (both these were my ideas incidentally), the user data-Hunk standard and overlay (which though not my idea arose out of the need to have user definable Direction Pointer Tables for TEL's MAP function) and the user transparent storage media handling. The multiple PARSE has been available but undocumented from version A10, it became a documented feature when the other three features were added at version A1a.

The user overlays provide for additional functions to be added to the editor, there are four at present, PAW-TEL, PAW-PHOSIS and MEGA in the PTM user overlay add-on, and overlay H the data-Hunk management overlay which is supplied with PAW. Overlay Z is reserved for user's user overlays, the letters K, U and W have also been 'booked'.

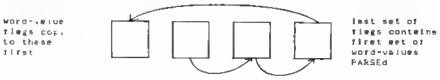
Prior to the modification to allow multiple use of PARSE, only the first logical sentence could be extracted, this meant you could not give more than one command at a time to other characters. Now it is possible to extract many sentences from text within double-quotes. However, unlike player input these must all be extracted in one time-frame, which requires that each sentence in turn is extracted and stored before extracting the next one or returning to deal with direct player commands. There is also a minor difficulty of acting upon multiple commands to other characters synchronously with player actions - it's no good if the commanded character performs all the actions he is instructed to do within one time-frame!

To store multiple PARSEd sentences requires the word-values in flags 33-36 and 43-47 (the word-flags) to be moved to other flags with COPYFF before performing another PARSE. To make this work usefully for more than two sentences (ie with one stored and one still in the word-flags) requires the use of a shift-register setup, with the stored values in flags being moved-on to yet other flags before the word-flags are stored in them.

To act on these synchronously requires the first sentence stored to be moved back into the word-flags, and for each following sentence likewise in each subsequent time-frame until all are dealt with. The shift-register storage is used in what is termed a ring-register mode.


word-value flags copy these last
last set of storage flags copy to last set first

Repeat after each PARSE and then do the following during a time-frame in which multiple PARSE is performed on for subsequent time-frames until all stored sentences are dealt with.


word-value flags copy to these first
last set of flags contains first set of word-values PARSEd

The user data-Hunk facility has many uses, it's primary function is to allow data for user overlays to be stored in a controlled and consistent manner within a database. It can also be used to include such things as a screen-dump routine for the Spectrum +3, or dot-matrix printer screen-dumps for 48K users. These however must be produced in position independent code as a Hunk may move, unless that is, it is installed each time as the first Hunk and no character sets are added.

You can also incorporate extra data to be used by EXTERNS. one idea I had was for an AUTOEXAmine function, with a table containing a number for each object which would be the number of the message containing additional info on each object or that of a "nothing special" message.

The user transparent storage media (tape/disk/microdrive cartridge) handling means that anyone producing user overlays does not (except in the case of verify) concern themselves as to what media is being used for loading and saving as they are all dealt with in a consistent manner.

There are handlers for tape and +3 disk, I'm fairly certain there is one for +D/Disciple users, the one for OPUS disk was dropped when this disk-drive was discontinued. There should have been one for Microdrive, but as I was promised a copy of this version when it became available and no such thing has been forthcoming I can only conclude it was never released.

The only other PAW currently in existence is a text-only CP/M one, for CPC, PCW 8256/8512 and Spectrum +3 of course, and any other 3-inch disk-using CP/M machines (the only one that springs to mind is the Tatung Einstein). The originally advertised CPC version was as far as I know never started, and the Commodore 64 version although started has, it seems, been abandoned in favour of 'PAW Espania' (the Spanish language version of PAW) and PCPAW (perhaps.)

As PAW-PHOSIS allows PAW users to build up a library of useful sub-processes, would there be any interest in having a central repository (or swap-shop) for the mutual collection and exchange of these? (Strictly user's own productions and no 'borrowings' from commercial games). Basically these would be compiled onto a tape and all contributers would obtain a copy for the cost of materials, postage and packing, and handling. Any takers?

Useful addresses

If you have any other addresses you've found useful in the past, let me know and I'll include them in future issues.

AMI = Amiga	ELE = Electron
ARC = Archimedes	S48 = Spectrum range
BBC = Acorn BBC Micro	ST = Atari ST range
C64 = Commodore 64/128	VAR = various computers
CPC = Amstrad CPC range	

OTHER ADVENTURE MAGAZINES

VAR: Claus Nygaard, Adventure Posten, Adventure Klubben, Vestergade 25A, 4930 Maribo, Danmark.

VAR: Mandy Rodrigues, Adventure Probe, 24 Maes Y Cwm, Llandudno, Gwynedd, LL30 1JE.

S48 tape: Magic Missile, Futuresoft, 75 Ben Rhydding Road, Ilkley, West Yorkshire, LS29 8RN.

VAR: Mike Brailsford, Spellbreaker, 19 Napier Place, South Parks, Glenrothes, Fife, KY6 1DX.

ST disk: Syntex, 9 Warwick Road, Sidcup, Kent, DA14 6LJ.

ADVENTURE COLUMNISTS

VAR: Steve Cooke, Ace, Priory Court, 30-32 Farringdon Lane, London, EC1 3AU.

AMI: Dave Eriksson, Amiga Computing, Database Publications Ltd, Europa House, Adlington Park, Adlington, Macclesfield, SK10 4NP.

CPC: The Pilgrim, Amstrad Action, Future Publishing Ltd, 4 Queen Street, Bath, BA1 1EJ.

ST: Brillig, Atari ST User, Database Publications Ltd, Europa House, Adlington Park, Adlington, Macclesfield, SK10 4NP.

C64: Andy Moss, Commodore Computing International, Croftward Ltd, Finsbury Business Centre, 40 Bowling Green Lane, London, EC1R 0NE.

C64: Gordon Hamlett, Commodore Disk User, Argus Specialist Publications Ltd, Argus House, Boundary Way, Hemel Hempstead, HP2 7ST.

C64/AMI: Keith Campbell, Commodore User, Priory Court, 30-32 Farringdon Lane, London, EC1 3AU.

VAR: Keith Campbell, Computer + Video Games, Priory Court, 30-32 Farringdon Lane, London, EC1 3AU.

ELE: Pendragon, Electron User, Database Publications Ltd, Europa House, Adlington Park, Adlington, Macclesfield, SK10 4NP.

VAR: Paul Rigby, The Games Machine, PO Box 10, Ludlow, Shropshire, SY8 1DB.

BBC: The Mad Hatter, The Micro User, Database Publications Ltd, Europa House, Adlington Park, Adlington, Macclesfield, SK10 4NP.

VAR: Tony Bridge, Popular Computing Weekly, Greencoat House, Francis Street, London, SW1P 1DG.

S48: The Sorceress, Sinclair User, Priory Court, 30-32 Farringdon Lane, London, EC1 3AU.

S48: Mike Gerrard, Your Sinclair, 14 Rathbone Place, London, W1P 1DE.

C64/AMI: Prof Norman Nutz, ZZAP!, PO Box 10, Ludlow, Shropshire, SY8 1DB.

ADVENTURE COMPANIES

VAR: Alternative Software Ltd, Units 3-6, Baileygate Industrial Estate, Pontefract, West Yorkshire, WF8 2LN. Telex: 557994 RR DIST G Fax: (0977) 790243 Tel: (0977) 797777

VAR: Digital Dynamite, 54 Watermill Road, Fraserburgh, Grampian, Scotland, AB4 5RJ.

VAR: Mitch Pomfret, M.S.B Games, 2 Bude Close, Bramhall, Stockport, Cheshire, SK7 2QP. (GAC)

VAR:(no utilities!) Mastertronic, 2-4 Vernon Yard, Portobello Road, London, W11 2DX.

VAR: Rack-It, Hewson Consultants Ltd, 56B Milton Park, Abingdon, Oxon, OX14 4RX. Tel: (0235) 832939

S48: John Wilson, Zenobi Software, 26 Spotland Tops, Cutgate, Rochdale, Lancashire, OL12 7NX.

ADVENTURE UTILITIES AND/OR ADD-ONS

S48: Camel Micros, Wellpark, Willeys Avenue, Exeter, Devon, EX2 8BE.

CPC: Roger Bankin, Graduate Software, 14 Forrester Avenue, Weston on Trent, Derbyshire, DE7 2HX.

VAR: Incentive Software Ltd, Zephyr One, Calleva Park, Aldermaston, Berkshire, RG7 4QW. Tel: (07356) 77288 Fax: (07356) 6940

VAR: Gilsoft International Ltd, 2 Park Crescent, Barry, South Glamorgan, CF6 8HD. Tel: (0446) 732765

S48: Gerald Kellett, Kelsoft, 28 Queen Street, Stamford, Lincolnshire, PE9 1QS.

CASSETTE DUPLICATORS

JBS Records, Freepost, 19 Sadlers Way, Hertford, SG14 2BR.

McGregor Tape Services, 42 Anchor Avenue, Paisley, PA1 1LD.

Simon Stable Productions, 20 West End, Launton, Oxon, OX6 0DF.

STATIONERY, PACKAGING AND PRINTING

Launton Press Ltd, Wedgewood Road, Bicester, Oxon.

Millway, Chapel Hill, Stanstead, Essex.

S&M (Processing) Ltd, Gotts Road, Wellington Bridge, Leeds, LE12 1ES.

* * * * * FANTASY AND SCIENCE FICTION ADVENTURES PRESENT * * * * *

MAGNETIC MOON

In this 3 part, text only, science fiction adventure, your spaceship has been dragged down onto a strange moon by a tractor beam, and now lies in the grip of a magnetic field, unable to take off. Your captain calls for volunteers to search for the source of the magnetic field, and you are one of the first to step forward. To your horror, the captain refuses to let you go, saying that you are needed on board to help the 1st Lieutenant supervise repairs! You want to get in on the action, and decide to jump ship and search for the source of the magnetic field yourself! However, first you must get off the ship without the captain or the 1st lieutenant catching you....!

MAGNETIC MOON is available in 48k and 128k versions, both at £3.49 - but see below for a special offer!

* * * * * * * * *

AVAILABLE SOON!! The long-awaited, 3 part sequel to "Magnetic Moon"............

STARSHIP QUEST

After freeing the "Stellar Queen" from the grip of the "Magnetic Moon", the spaceship is now heading for a hyperspace jump to Rigel III, in order to get the injured Commander Giles, who you rescued from the wrecked "Pathfinder", to proper medical facilities. The commander has given you information that leads you to believe that the secret of the two discs that the beautiful priestess Jaalsina gave you - the "Keys to the Universe" - may be found on the planet the "magnetic moon" orbits. Your captain says it is impossible for the ship to stop or turn back so that you can return to the mother planet, so you realise that you will have to go it alone again, and jump ship! But you have only TWO MINUTES before the ship goes into hyperspace! Can you find some equipment and get off the ship in time? What perils await you on the abandoned planet?? Play STARSHIP QUEST and find out!!

STARSHIP QUEST will be available, in 48k and 128k versions, at the beginning of October. The 128k version has more puzzles and locations, plus HELP messages in many locations. Price of both versions will be £3.99, but you can order your copy in advance for only £3.49! SPECIAL OFFER!!!! Order a copy of MAGNETIC MOON at the same time and you can have the two for only £5.99!!! Note that this offer will close on October 30th, so order both of these adventures NOW!!

Send your cheque/PO, stating which Spectrum computer you have, to:-

FSF ADVENTURES, 40 Harvey Gardens, Charlton, London, SE7 8AJ

Remember:
MAGNETIC MOON only = £3.49 STARSHIP QUEST (advance order) only = £3.49
MAGNETIC MOON and STARSHIP QUEST = £5.99

Utilities available

If you know of any other utilities or add-ons, especially for computers such as the MSX and Atari 8-bits, whatever, please write in and help make this list a definitive guide.

AMI = Amiga	DRG = Dragon 32
CPC = Amstrad CPC range	ELE = Electron
ARC = Archimedes	MTR = Master
BBC = Acorn BBC Micro	S48 = Spectrum 48k
C64 = Commodore 64	ST = Atari ST range

PROGRAM NAME	COMPANY (COMPUTERS) COMMENT
A-CODE	Level 9 (many) In-house utility only
ADLAN	Graduate (CPC)
ADVENTURE BUILDER SYSTEM	M A Richards (S48)
ADVENTURE CONSTRUCTION SET	Electronic Arts (C64)
ADVENTURE KERNEL SYSTEM	Melbourne House (AMS) book listing/tape
ADVENTURE WRITER	Codewriter (C64) USA Quill
ADVENTURESCAPE	A&B (BBC)
ADL	Public Domain (AMI)
ADVSYS	Public Domain (ST)
ALPS	Alpine Software (BBC MTR ARC)
AMIGAC?	Incentive (AMI) Where is it?
AMIGAVENTURE	Public Domain (AMI)
THE BIRO	Kemjem Corporation (many) In-house/to loan
CHARACTER SETS	Sinicro (S48) GAC
CHARACTERS	Gilsoft (S48) Quill
DRAGON WRITER	Cowen (DRG)
DUNGEON BUILDER	Dream (C64)
THE EXPANDER	Gilsoft (S48) with PRESS
FONT CREATOR	Sinicro (S48) GAC
THE FIX	Kelsoft (S48) Quill
THE FIX+	Kelsoft (S48) Quill - unreleased
GAC	Incentive (S48 AMS C64)
GAC+	Incentive (C64) disk-only
GAC DATABASE PRINTER	Big Sky (C64)
THE GACPAC	Essential Myth (S48) GAC
GENESIS	CRL/Camel Micros (S48) good band!
THE ILLUSTRATOR	Gilsoft (S48 AMS C64) Quill
MEGA	Gilsoft/Kelsoft (S48) PAW, part of PTM
MINIFIX	Kelsoft (S48) Quill
PATCH	Gilsoft (S48) Quill
PAW	Gilsoft (S48 AMS PC) no C64/ST!
PAW-PHOSIS	Gilsoft/Kelsoft (S48) PAW, part of PTM)
PAW-TEL	Gilsoft/Kelsoft (S48) PAW, part of PTM)
PIC-FIX	Kelsoft (S48) Quill
PRESS	Gilsoft (S48) Quill
PTM	Gilsoft,Kelsoft (S48) 3 PAW overlays
QUAID	Kelsoft (S48) Quill
THE QUILL	Gilsoft (S48 AMS C64)
RECLAIMER	Kelsoft (S48) GAC
SAGA	Scott Adams (C64) not for sale'
THE SCRIBE	Your Spectrum (S48) listing
STAC	Incentive (ST)
TAC	Incentive (BBC ELE) GAC without graphics
TAILSPIN	Microdeal (ST AMI)